PATENTTI- JA R[illegible]STERIHALLITUS
NATIONAL BOARD OF PATENTS AND REGISTRATION

PCT/FI00/00512

Helsinki 8.9.2000

REC'D 03 OCT 2000
WIPO PCT

4

ETUOIKEUSTODISTUS
PRIORITY DOCUMENT

Hakija Applicant	**Nokia Corporation** **Helsinki**
Patenttihakemus nro Patent application no	**991492**
Tekemispäivä Filing date	**30.06.1999**
Kansainvälinen luokka International class	**H04L**
Keksinnön nimitys Title of invention	

"Bearer adapter management at a gateway server"
(Siirtosovittimen hallinta yhdyskäytäväpalvelimella)

Täten todistetaan, että oheiset asiakirjat ovat tarkkoja jäljennöksiä patentti- ja rekisterihallitukselle alkuaan annetuista selityksestä, patenttivaatimuksista, tiivistelmästä ja piirustuksista.

This is to certify that the annexed documents are true copies of the description, claims, abstract and drawings originally filed with the Finnish Patent Office.

[illegible]
Eija Solja
Apulaistarkastaja

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Maksu	300,-	mk
Fee	300,-	FIM

L 3

Bearer adapter management at a gateway server

The present invention relates to management of bearer adapters at a gateway server. It is particularly suitable for a mobile protocol such as WAP (Wireless Application Protocol) for enabling a mobile terminal to access the Internet.

The term "Internet" is commonly used to describe information, content, which can be accessed using a terminal, typically a PC, connected via a modem to a telecommunications network. The content can be stored at many different sites remote from the accessing computer, although each of the remote sites is also linked to the telecommunications network. The content can be structured using Hypertext Mark-up Language (HTML). The Internet is made workable by the specification of a standard communications system which makes use of a number of protocols, such as the Transfer Control Protocol (TCP), the User Datagram Protocol (UDP), and the Internet Protocol (IP), to control the flow of data around the numerous different components of the Internet. TCP and UDP are concerned with the prevention and correction of errors in transmitted Internet data. IP is concerned with the structuring and routing of data. On top of that, other application specific protocols may be provided to manage and manipulate the various kinds of information available via the Internet, for example HTTP to access HTML content, FTP to access files or SMTP to access e-mail.

The Internet is physically constructed from a hierarchy of telecommunication and data communication networks, for example local area networks (LANs), regional telephone networks, and international telephone networks. These networks are connected internally and externally by so-called "routers" which receive data from a source host, or a previous router in a transmission chain, and route it to the destination host or the next router in the transmission chain.

With increased use of mobile cellular telephones, there is a growing demand for so-called mobile Internet access, in which access is made from a portable computer connected to a cellular telephone or from an integrated computer/cellular phone device. Typically, the purpose of such access is to obtain content from the Internet. It has also been proposed to provide Internet access to advanced mobile terminals, so-called communicators and smart phones, by means of the Wireless Application Protocol (WAP), for example. WAP has an architecture in which there is a protocol stack having an application layer (called the Wireless Application Environment or WAE), a session layer (called the Wireless Session Protocol or WSP), a transaction layer (called the Wireless Transaction Protocol or WTP), a security layer (called Wireless Transport Layer Security or WTLS) and a transport layer (called the Wireless Datagram Protocol or WDP) as shown in Figure 1. Each of the layers of the architecture is accessible by the layers above as well as by other services and applications. These protocols are designed to operate over a variety of different bearer services such as SMS (Short Message Service), CSD (Circuit Switched Data), GPRS (General Packet Radio Service) etc. A specification describing the WAP architecture and the protocol layers is available from http//www.wapforum.org/.

At the above URL address one of the WAP specifications that can be found is the Wireless Datagram Protocol specification, i.e. the WDP specification. It specifies that between the WAP stack and bearers there is an Adaptation Layer. The Adaptation Layer is the layer of the WDP protocol that maps the WDP protocol functions directly onto a specific bearer. The Adaptation Layer is different for each bearer and deals with the specific capabilities and characteristics of that bearer service. Moreover, at the WAP Gateway or server the Adaptation Layer is also called a Tunnel that terminates and passes the WDP packets on to a WAP Proxy/Server via a Tunnelling protocol, which is the interface between the Gateway that supports the bearer service and the WAP Proxy/Server.

The Adaptation Layer or Bearer Adapter as it will be called in this document is thus a component that connects the WAP Server to the wireless network. To support a number of different bearers the gateway server will thus need to have a number of different bearer adapters. New bearers become available as networks develop. For example GSM GPRS is not yet in use but is estimated to be taken into use within one or two years. Also the third generation systems are estimated to be taken into use within two to five years. Thus operators and companies holding gateway servers, such as WAP gateways are likely to need to update the server to support new bearers. Also a gateway might be taken into use with one bearer only to start with, and then add other bearers to compliment the range by servicing different customers (i.e. terminals supporting a particular but different bearer). The protocol stack (in WAP the WAP stack) needs to support each bearer adapter.

Now a gateway has been invented where bearer adapters are managed dynamically, thus allowing adding new bearer adapters dynamically, preferably both after the gateway server has been installed and also while it is able to communicate with other, already existing bearer adapters. Also according to the present invention the gateway server has been arranged to enable deleting bearer adapters dynamically both after installation of the gateway server and while the gateway server is able to communicate with other, still existing bearer adapters.

It is advantageous to allow adding and/or deleting bearer adapters while the gateway server is able to communicate with existing bearer adapters as that way bearer adapters can be managed without interrupting the operation of the gateway server. Thereby bearer adapters can be added without rebooting the server.

In a preferred embodiment of the invention the dynamic addition of bearer adapters is implemented by creating at the protocol stack an own new thread for each bearer adapter. This way the protocol stack supports the new bearer adapter and there is no need to stop the gateway server in order to reconfigure the protocol stack. The dynamic deletion of bearer adapters is implemented by introducing a bearer gate inbetween the bearer adapter and the protocol stack, whereby the deletion of a bearer adapter leads to deletion of the specific bearer adapter from the bearer gate memory, although in the particular embodiment the thread remains at the protocol stack until the gateway server is shut down next time. The bearer gate watches that the protocol stack will not try to send anything to a deleted bearer adapter.

Further management and control of the bearer adapters is simplified by a graphical user interface allowing an administrator to both dynamically add and delete bearer adapters by simple selections made with the graphical user interface.

According to a first aspect of the invention there is provided a server for managing bearer adapters, each bearer adapter being used at a server for communication with a terminal over a particular wireless network, the server comprising:
means for dynamically adding a bearer adapter to the server while the server is able to communicate with already existing bearer adapters.

In one particular embodiment, the invention comprises a gateway server serving a plurality of mobile terminals. It may be a WAP gateway. For example, commands, such as WAP requests, may be sent in short messages (generated by SMS) and sent to a WAP/HTTP gateway. The gateway will interpret these as WAP network packets and will perform the necessary HTTP transactions on an origin server. After that it sends back a WAP message on the same bearer, i.e. as an SMS message containing the result.

In another particular embodiment, the server comprises creating means for creating a thread in response to adding a bearer adapter, and assigning means for assigning the created thread to the added bearer adapter.

According to a second aspect of the invention there is provided a method of managing bearer adapters, each bearer adapter being used at a server for communication with a terminal over a particular wireless network, the method comprising:
dynamically adding a bearer adapter to the server while the server is able to communicate with already existing bearer adapters.

A bearer adapter is added by creating a particular thread to which the added bearer adapter is assigned. More particularly the thread is created at the wireless protocol stack. Further the method according to the invention comprises

In the following example, communication is described with reference to the Wireless Application Protocol (WAP) mentioned above. It should be noted that the invention is not limited to the use of WAP and other protocols and specifications may be used.

Figure 2 shows a communication system comprising a plurality of mobile terminals 2 having access to the Internet 4. The mobile terminals transmit signals 6 which are received by and transmitted through a wireless network 8. The wireless network can be a number of different network systems such as GSM, CDMA IS-95, TDMA IS-136, and UMTS, and can use different type of communication within one and the same system, for example SMS, GPRS or HSCSD communication within GSM. Accordingly a number of different bearers can be used for transmitting signals 6. WAP requests 6 received by the network 8 are routed to a proxy or gateway server 12. The server 12 translates WAP requests into HTTP requests and thus allows the mobile terminals 2 to request information from a web server 14 and thus browse the Internet 4. Information obtained from the web server 14 is encoded by the proxy into a suitable format and then transmitted by the wireless network to the mobile terminal 2 which requested it. The response comprises wireless mark-up language (WML) according to WAP. WML is a tag-based display language providing navigational support, data input, hyperlinks, text and image presentation, and forms. It is a browsing language similar to HMTL. The mobile terminal 2 processes and uses the information. If the web server 14 provides content in WAP/WML format, the server 12 can retrieve such content directly from the web server 14. However, if the web server provides content in WWW format (such as HTML), a filter may be used to translate the content from WWW format to WAP/WML format.

The Wireless Application Protocol is applicable to a number of different systems including GSM-900, GSM-1800, GSM-1900, CDMA IS-95, TDMA IS-136, wide-band IS-95 and third generation systems such as IMT-2000, UMTS and W-CDMA.

Although Figure 2 shows information being obtained from the Internet, the proxy itself may contain the desired information. For example, the client may retrieve information from the file system of the proxy.

In addition to the web server 14, the mobile terminals may communicate with a wireless telephony application (WTA) server 18. Also other types of origin servers are possible.

Figure 3 shows a gateway server embodied in hardware such as a computer 20. The computer 20 has dynamic memory, processing power and memory to store all of the programs needed to implement the gateway server such as the application program, the protocol stacks and the operating system. The computer 20 comprises a user interface such as a keyboard 22 and a display 23 and a server program 24. The server program 24 has an application program 26 for processing events of the underlying protocol, such as handling a request to retrieve WML from a server, and protocol stacks such as a WAP protocol stack 28 and a HTTP protocol stack 30. The application program 26 controls flow of data, including commands, requests and information, between the computer and various networks including a telephone network 32, the Internet 34 and a data network and circuit switched data networks 35. The application program 26 may further run a program that can be seen on the display 23 and controlled with the keypad 22 (and e.g. a mouse). The computer 20 communicates with the Internet 34 through the HTTP protocol stack 30 and an interface 36. The computer 20 communicates with the telephone network 34 and the data network 35 through interfaces 38 and 40. The server program 24 also comprises a gateway 42 which converts between HTTP and WAP. SMS messaging may be provided via a data connection through appropriate hardware to the operator's network.

Individual threads 44 present in the application program 26 and the WAP protocol stack 28 use processors 46 in the computer 20 to carry out necessary processing tasks. Allocation of threads to processors is provided by threading services 48 present within the operating system 50 of the computer 20.

As shown in Figure 1 the WAP stack is built on top of so called bearers (which provide datagram services). These bearers can be, for example, SMS or CSD. The bearers have their own protocol and are implemented through protocol stack implementations.

Figure 4 shows a functional block diagram (embodied in software) of a gateway server according to the present invention, at least to the extent for understanding the invention. The gateway server includes a Wireless Protocol Stack (WPS) 50, such as the WAP stack shown in Figure 1. Below the WPS are the different bearer adapters 51 which access the different bearers through bearer drivers 52. Now there is provided between the WPS and the bearer adapters a bearer gate 53, which isolates the WPS from the bearers and controls the starting and stopping of datagram traffic between a bearer adapter and the WPS. The bearer gate 53 further has a link to a bearer manager 54, which controls and configures the bearer adapter operation. The Bearer Manager 54 gets control commands from the administrator 55, who is allowed to control bearer adapter operation with a user interface 56, such as the keypad 22 and display 23 shown in Figure 3. The connection to Internet, such as to a web server is via interface 57.

The gateway server uses the bearer gate 53 and bearer adapter 51 in two ways:
1) To transmit data to a particular wireless network,
2) To control and monitor the bearer operation.
Between the bearer gate 53 and WPS 50 there is an interface 58a, which here will be called I_WDPBI, which is an interface to send and receive WDP datagrams and to retrieve information about the Bearer adapter 51. Further the datagrams are transferred between the bearer gate and the bearer adapter over interface 58b. Thereby the interface implementing the above mentioned point 1) is established by interfaces 58a and 58b. There is further an interface 59 between the bearer manager 54 and bearer gate 53 for controlling and configuring the operation of the bearer adapter 51. This interface 59 is called I_BGM, and accordingly implements the above mentioned point 2). Via the User Interface 56 bearer adapters can be added, removed, controlled, configured and monitored.

The different operations and functional blocks shown in Figure 4 are preferably implemented as software blocks, which are run by processor 46 by calling threads 44 in the protocol stack 28 and in the application program 26. The threads in relation to the bearer adapters 51 are shown more closely in Figure 5.

All services in interface 59 (I_BAM) are called in a single management thread context, *MgmtCntx* 61, which is a thread in the server application program 26. *I_WDPBI* services, i.e. services over interface 58 will be called by two threads from the WPS (with the aid of the bearer gate). There is one thread at the WPS, *SendContext* 62, for sending data from the WPS and for controlling bearer operation. In sending the thread *SendContext* 62 retrieves a datagram from a buffer at the WPS 50 and sends it with a bearer, whose identification the datagram contains, and then retrieves the following datagram from the buffer. A datagram is thus only sent to one bearer at a time. Adding or removing bearer adapters does therefore not disturb the function of the thread *SendContext* 62, who will only realise the adding or removal from the fact that datagrams go to different bearer adapters than before. Similarly the management thread, *MgmtCntx* 61 only has calls for one bearer at a time, and thus adding or removing bearer adapters while the server is able to communicate with existing bearer adapters, does not disturb the function of the management thread. The other thread at the WPS, *RecvContext* 63, 64, is receiving data from the bearer adapter 51. In creating a new Bearer adapter 51 the thread *SendContext* 62 operates initialisation functions between the WPS and bearer gate, and there is a blocking call from the thread *RecvContext* 63, 64.

Each instantiated bearer adapter 51 shares the threads *MgmtCntx* 61 and *SendCntx* 62 and each instance has its own thread *recvCntx*, which is created at the WPS when a bearer adapter is created. This is shown by having thread, *recvCntx1* 63, for a first bearer adapter BA1 and having another thread, *recvCntx2* 64, for a second bearer adapter BA2. The fact of assigning or creating an own thread *recvCntx* in the WAP protocol stack 50 for each bearer adapter 51 allows dynamic creation of bearer adapters while the gateway server is able to communicate with existing bearer adapters. This is since the server can not control when it has something to receive, i.e. data can come from two different bearers at the same time. Therefore having an own thread for each bearer for reception guarantees smooth operation of the server. In the preferred embodiment a new thread 44 (Fig. 3) is created (*recvCntx)* at the protocol stack 50 (reference number 28 in Fig. 3) when a command is received to create a new bearer adapter 51. When attaching a bearer adapter to the WPS 50, a bearer adapter identification is given as a field in bearer description structure, which is additionally held at the bearer gate 53. The WPS passes the identification as a parameter in every function call through the interface 58. By creating a new thread for a new bearer adapter while the server is able to communicate with existing bearer adapters, there is no need to reboot the server in order to have this new bearer adapter installed at the protcol stack, and thereby the server operation does not need to be interrupted.

In following threads are explained to help understand the invention. A thread is basically a path of execution through a program and can be the smallest unit of execution that is scheduled on a processor. A thread consists of a stack, the state of the CPU registers, and an entry in the execution list of the system scheduler.

A thread is a single sequential flow of execution in program code and has a single point of execution. To deal with a simple process, a program comprising a single thread can be used. For more complex processes which involve running a number of applications, a program can rely on a number of threads. Operating systems usually provide thread management for the application (creation, termination and specifying the entry point: at the start of the program code).

A process consists of one or more threads and the code, data, and other resources of a program in memory. Typical program resources are open files, semaphores, and dynamically allocated memory. Each thread shares all of the process resources of the process. A program executes when the system scheduler gives one of its threads execution control. The scheduler determines which threads should run and when they should run. Threads of lower priority may have to wait while higher priority threads complete their tasks. On multiprocessor machines, the scheduler can move individual threads to different processors to "balance" the load on the central processing unit.

Each thread in a process operates independently. Unless they are made visible to each other, the threads execute individually and are unaware of the other threads in a process. Threads sharing common resources, however, must co-ordinate their work, for example by using semaphores or another method of inter-process communication.

Dynamic bearer deletion has been enabled by introducing a bearer gate 53 between the WPS 50 and bearer adapters 51 for isolating the WPS from the bearers. When a command comes from the UI 56 to the bearer manager 54 to remove a bearer adapter, that particular bearer adapter is removed from the bearer gate 53. In that sense the bearer gate keeps a list, i.e. stores in memory information about each bearer adapter. The thread *recvCntx* 63, 64 for that particular bearer adapter remains at the WPS until the server is stopped. However, during that time if the WPS tries to send something to the removed bearer adapter, the bearer gate returns an error message.

The gateway server can simultaneously contain multiple bearer adapters 51 for the same or a different wireless network. Thereby there can be two different bearer adapters for SMS messages, or alternatively the same bearer adapter could be used for sending short messages through two different SM-SCs (Short Message Service Center).

The bearer control operations for dynamically controlling the bearer adapters has further been enhanced by a user interface 56 for the administrator 55 of the gateway server. Accordingly the gateway server according to the present invention is provided with a user interface allowing the administrator to dynamically add new bearers while the server is able to communicate with bearer adapters already existing in the gateway. Preferably bearer adapters can be added, removed, controlled, configured and monitored with the user interface, which preferably comrises a graphical interface (on the display 23) with the aid of which the bearer adapter operation as well as the gateway server operation in whole can easily be controlled.

The graphical user interface is preferably windows based comprising one control window for installation, configuring, starting and stopping a bearer adapter, and another window which is a monitoring window for monitoring the operation of the bearer adapter, its statistics and log information. Alternatively there could be a third window for the log information only. The control window may include an icon for each bearer adapter, and by selecting one of the icons a bearer adapter management field is opened as shown in Figure 6a. The administrator 55 creates a new instance of a bearer adapter with UI 56. In the creation the administrator inputs the name of the bearer adapter instance and selects the bearer adapter type from a list. After the creation, the administrator configures the bearer adapter instance unless the default settings (that have been stored in the gateway server upon installation) are acceptable. The server loads the new software dynamically and creates the bearer adapter instance by creating a new thread as has been explained above. After the creation, the state of the bearer adapter instance is 'stopped'. Figure 6a shows normal software buttons according to the windows systems for starting and stopping a bearer adapter (Start/Stop), for configuring a bearer adapter (Configure), for creating new bearer adapters (Create new...) and for removing bearer adapters (Remove).

A bearer adapter instance can be configured in the 'stopped' and 'running' state. A bearer adapter instance is configured by editing property strings of the bearer adapter instance. If the bearer adapter instance is in the 'running' state, a change in the value of a property may not become active immediately, but in the next startup of the bearer adapter instance. Regardless of its state, the server stores the new values of the properties. Figure 6b shows a sample of the configuration dialog in the bearer adapter management UI.

Thus creating and removing bearers dynamically has been simplified by the aid of a graphical user interface, which is simple to use by the administrator 55, and by which dynamic bearer adapter management is allowed while the gateway server is able to communicate with bearer adapters existing in the gateway server. With the aid of the graphical user interface an administrator can easily manage bearer adapters without the need to have skills in a computer programming language.

Figures 7a - d show signalling diagrams between Bearer manager 54, WPS 50, bearer gate 53 and bearer adapter 51 when creating, removing, starting and stopping a bearer adapter. The Figures 7a - d do not show signalling to the user interface, but show the operation when the commands create (7a), start (7b), stop (7c) and remove (7d) come to the bearer manager from the user interface.

Figure 7a shows a signalling diagrams when a bearer adapter is created. Starting from above the first signal shows the bearer manager configuring a new bearer adapter. Once that is completed the bearer gate is informed of a new bearer adapter. The bearer gate then creates a thread for at the WPS after which the bearer manager is informed of the added bearer adapter. After that datagram traffic can start using that newly added bearer. The *I_WDPBI.init* and *I_WDPBI.open* signals represent calling initialisation and datagram traffic opening events from the *SendContext* thread when the WPS is to send datagrams. Thereafter the *I_WDPBI.receiveBuffer* signal represents a blocking call from the *RecvContext* thread.

Figure 7b shows a signalling diagrams when a bearer adapter is started. Starting from above the first signal shows the bearer manager starting a bearer adapter. Once that is completed the bearer gate is informed of starting the particular bearer adapter. The *I_WDPBI.init* and *I_WDPBI.open* signals represent calling initialisation and datagram traffic opening events from the *SendContext* thread, which came from the WPS when a new bearer adapter was created (in Fig. 7a) and which the bearer gate communicates to the bearer adapter when the adapter is started. The bearer gate then returns a call to the bearer manager informing that the particular bearer adapter has been started for datagram traffic. Thereafter the *I_WDPBI.receiveBuffer* signal represents a blocking call from the *RecvContext* thread, which came from the WPS when a new bearer adapter was created (in Fig. 7a) and which the bearer gate communicates to the bearer adapter when the adapter is started.

Figure 7c shows a signalling diagrams when a bearer adapter is stopped. Starting from above the first signal shows the bearer manager stopping a bearer adapter, whereby the bearer gate is informed of stopping the particular bearer adapter. The *WDPBI*.closeAll and *WDPBI.shutdown* signals represent events from the *SendContext* thread that are communicated from the bearer gate to the bearer adapter informing that the bearer adapter is stopped from sending. The bearer gate then returns a call to the bearer manager informing that the particular bearer adapter has been stopped. Thereafter the *receiveBuffer returns* event represents a blocking call from the *RecvContext* that is communicated from the bearer gate to the bearer adapter informing that the bearer adapter is stopped from receiving. The particular bearer adapter is then stopped from sending and receiving.

Figure 7d shows a signalling diagrams when a bearer adapter is removed. Starting from above the first signal shows the bearer manager removing a bearer adapter, whereby the bearer gate is informed of removing the particular bearer adapter. The bearer gate removes the particular bearer adapter from its memory and returns a call to the bearer manager informing that the particular bearer adapter has been removed. The bearer adapter is thus destroyed and the thread *RecvContext* that relates to the particular bearer adapter is destroyed next time the gateway server operation is stopped.

This paper presents the implementation and embodiments of the invention with the help of examples. It is obvious to a person skilled in the art, that the invention is not restricted to details of the embodiments presented above, and that the invention can be implemented in another embodiment without deviating from the characteristics of the invention. For example, although the foregoing is a description of mobile terminals browsing the Internet, it is to be understood that the communication may be of different types including sending and receiving information, conducting transactions such as financial transactions sending and receiving electronic mail or messages. The range of activities includes accessing services, for example weather reports, news, stock prices, flight schedules, downloading ringing tones, banking services including information provision and payments. It may occur in communications environments other than the Internet. Thus, the presented embodiments should be considered illustrative, but not restricting. Hence, the possibilities of implementing and using the invention are only restricted by the enclosed patent claims. Consequently, the various options of implementing the invention as determined by the claims, including the equivalent implementations, also belong to the scope of the present invention.

L4

Claims

1. A method of managing bearer adapters, each bearer adapter being used at a server for communication with a terminal over a particular wireless network, the method comprising:
dynamically adding a bearer adapter to the server while the server is able to communicate with already existing bearer adapters.

2. A method according to claim 1, wherein the method further comprises:
dynamically deleting a bearer adapter from the server while the server is able to communicate with still existing bearer adapters.

3. A method according to claim 1, wherein the method further comprises:
creating a particular thread to which the added bearer adapter is assigned.

4. A method according to claim 3, wherein the method further comprises:
creating said thread at a protocol stack in the server.

5. A method according to claim 1 and 2, wherein the method further comprises:
transferring data between a protocol stack and the bearer adapter via a bearer gate, and
upon creating the bearer adapter storing identification information about each bearer adapter in the bearer gate, and
upon deleting the bearer adapter removing the particular bearer adapter from the bearer gate.

6. A method according to claim 5, wherein the method further comprises:
upon deleting the bearer adapter keeping the particular thread assigned to it until the operation of the server is stopped next time.

7. A method according to claim 1, wherein the method further comprises:
controlling the operation of bearer adapters with a user interface.

8. A method according to claim 7, wherein the controlling comprises adding, removing, starting, stopping, configuring and monitoring the operation of bearer adapters.

9. A method according to claim 7 or 8, wherein the method further comprises:

controlling the operation of bearer adapters with a graphical windows based user interface.

10. A method according to any preceding claim in which the terminals comprise mobile terminals, for example cellular telephones, supporting the Wireless Application Protocol (WAP).

11. A server for managing bearer adapters, each bearer adapter (51) being used at a server for communication with a terminal over a particular wireless network (8), the server comprising:

means (53, 56, 63) for dynamically adding a bearer adapter (51) to the server while the server is able to communicate with already existing bearer adapters.

12. A server according to claim 11, wherein the server further comprises

a user interface (56, 22, 23) for allowing an administrator (55) of the server to dynamically add a bearer adapter while the server is able to communicate with already existing bearer adapters.

13. A server according to claim 11, wherein the server further comprises

creating means (50, 53) for creating a thread (63, 64) in response to adding a bearer adapter (51), and

assigning means (50, 53) for assigning the created thread (63, 64) to the added bearer adapter (51).

14. A server according to claim 11, wherein the server further comprises

a wireless protocol stack (50) for implementing a wireless protocol and for transferring data between the protocol stack and a bearer adapter (51), a bearer gate (53) for isolating the wireless protocol stack (50) from the bearer adapter (51) and for storing information on each bearer adapter.

15. A server according to claim 11, wherein the server further comprises

removing means (56, 54, 53) for dynamically removing a bearer adapter from the server while the server is able to communicate with still existing bearer adapters.

16. A server according to claim 14 and 15, wherein

the removing means have been arranged to remove the bearer adapter (51) from the bearer gate (53), and

the bearer gate (53) has been arranged to stop communication to the removed bearer adapter.

17. A server according to claim 12, wherein the user interface (56) further comprises a graphical windows based user interface.

18. A server according to any of claims 11-17 comprising a gateway server serving a plurality of mobile terminals.

19. A server according to claim 18 comprising a WAP gateway.

20. A computer program product for managing bearer adapters at a server, each bearer adapter being used at a server for communication with a terminal over a particular wireless network, the computer program product comprising:

computer readable program means (53, 56, 63) for dynamically adding a bearer adapter (51) to the server while the server is able to communicate with already existing bearer adapters.

Abstract

The invention relates to a gateway where bearer adapters are managed dynamically, thus allowing adding new bearer adapters dynamically while the gateway server is able to communicate with already existing bearer adapters. Also according to the present invention the gateway server has been arranged to enable deleting bearer adapters dynamically while the gateway server is able to communicate with still existing bearer adapters. The invention also relates to a method for managing bearer adapters and to a computer program product for managing bearer adapters at a server.

Fig. 6a

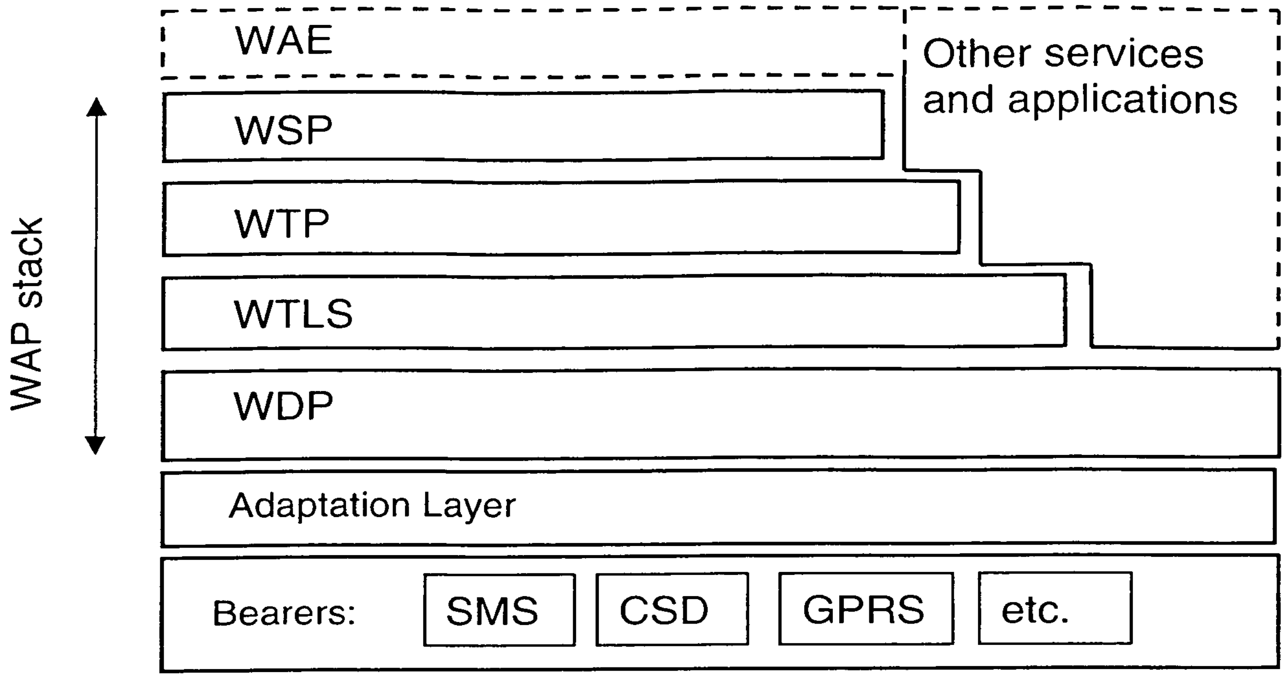

WAE
Other services and applications
WSP
WTP
WTLS
WDP
WAP stack
Adaptation Layer
Bearers:
SMS
CSD
GPRS
etc.

Fig. 1

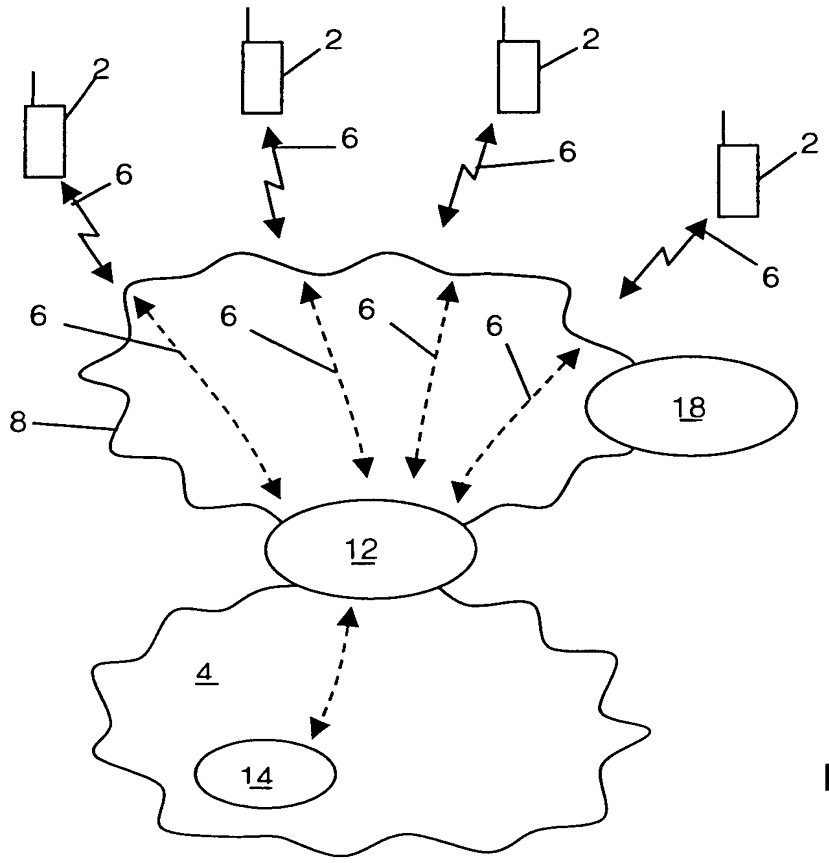

2
2
2
2
6
6
6
6
6
6
6
6
6
8
18
12
4
14

Fig. 2

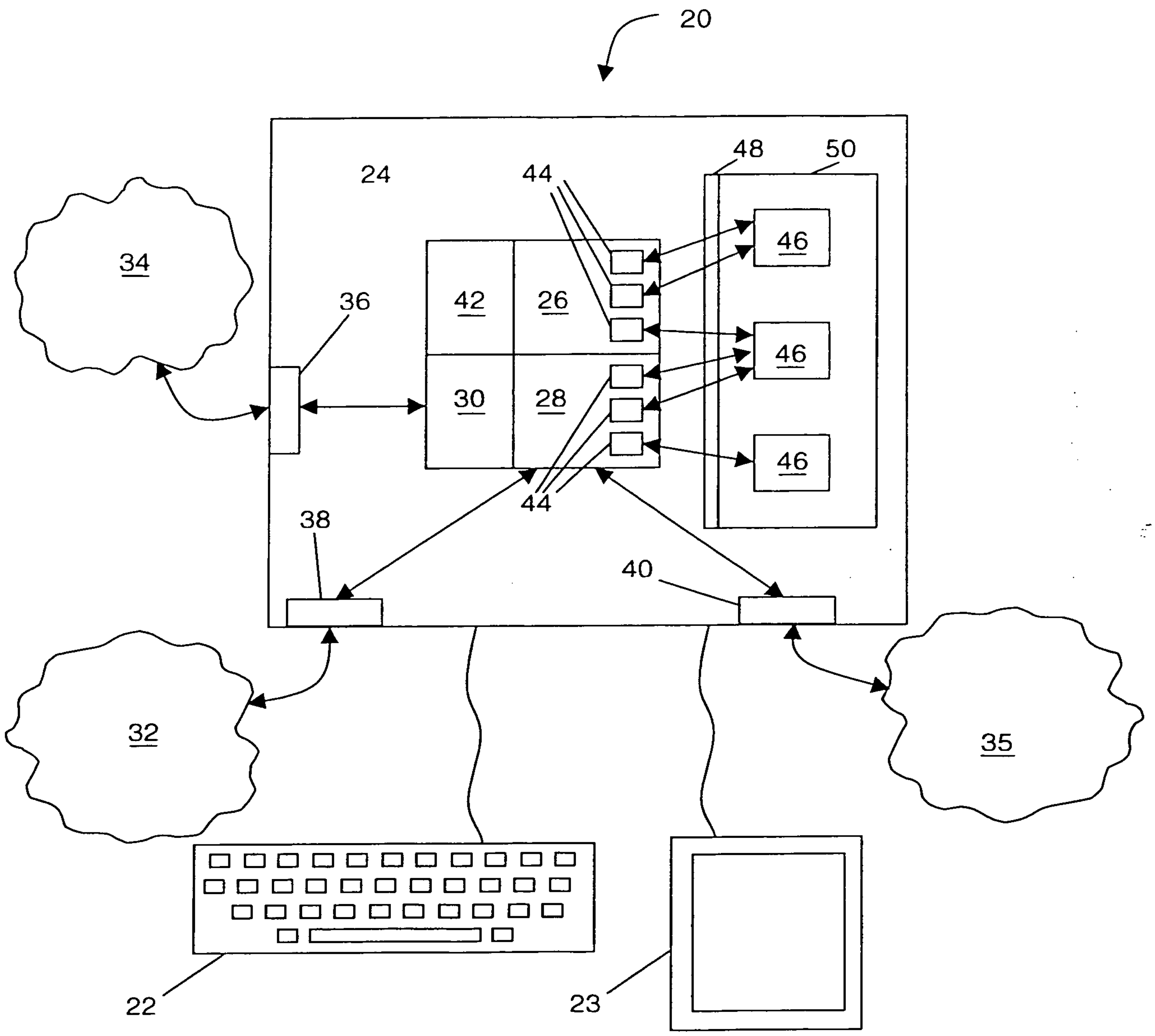
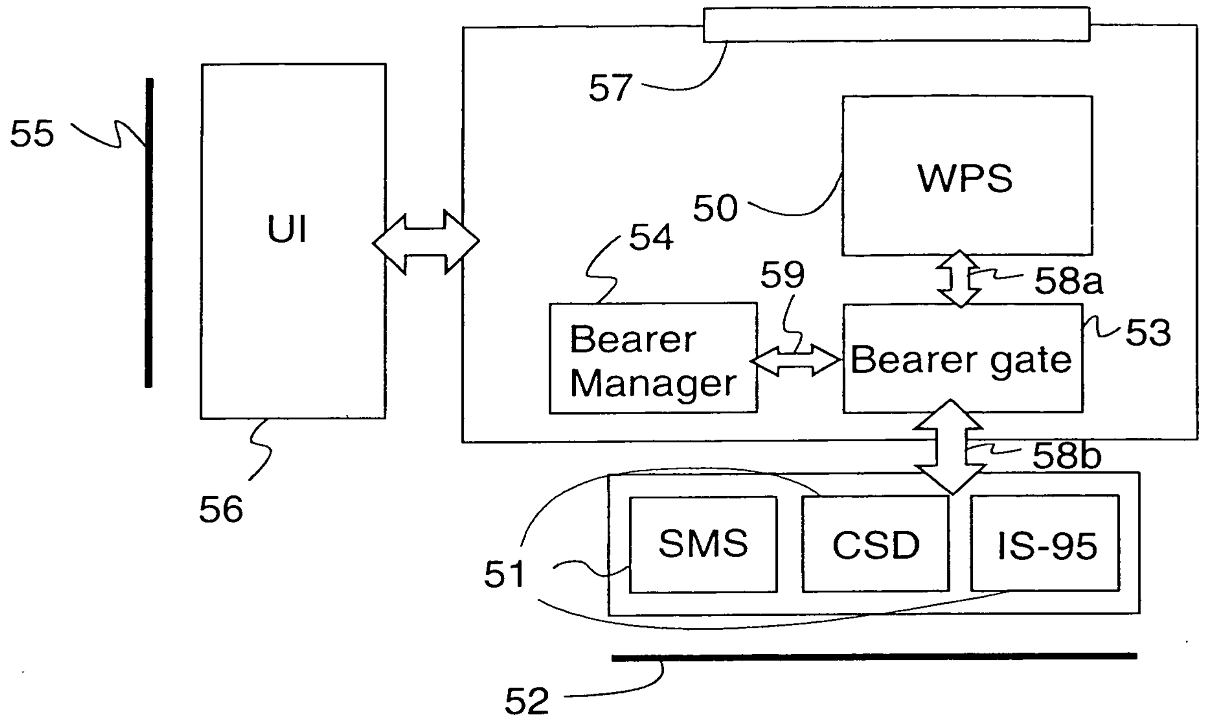
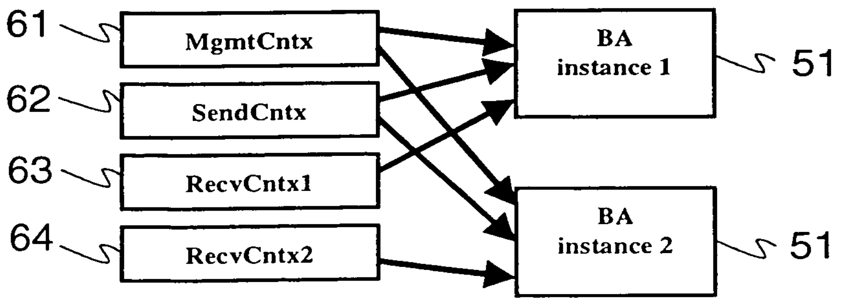


Fig. 3



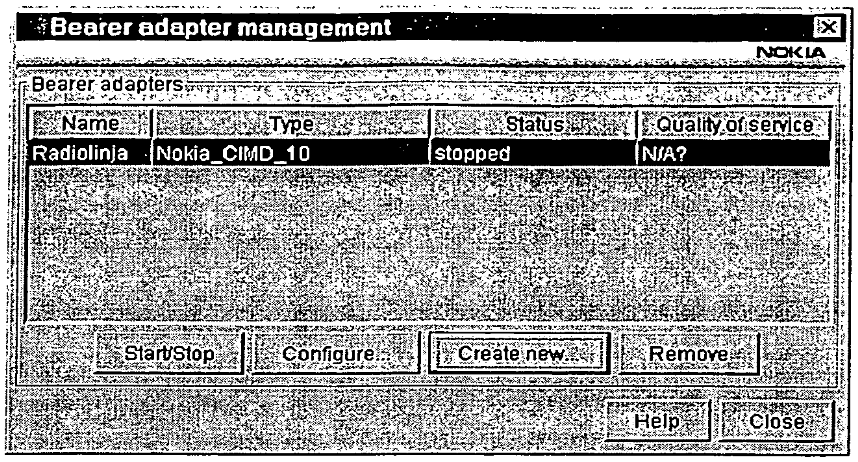
55
UI
56
57
WPS
50
54
59
58a
Bearer Manager
Bearer gate
53
58b
SMS
CSD
IS-95
51
52

Fig. 4



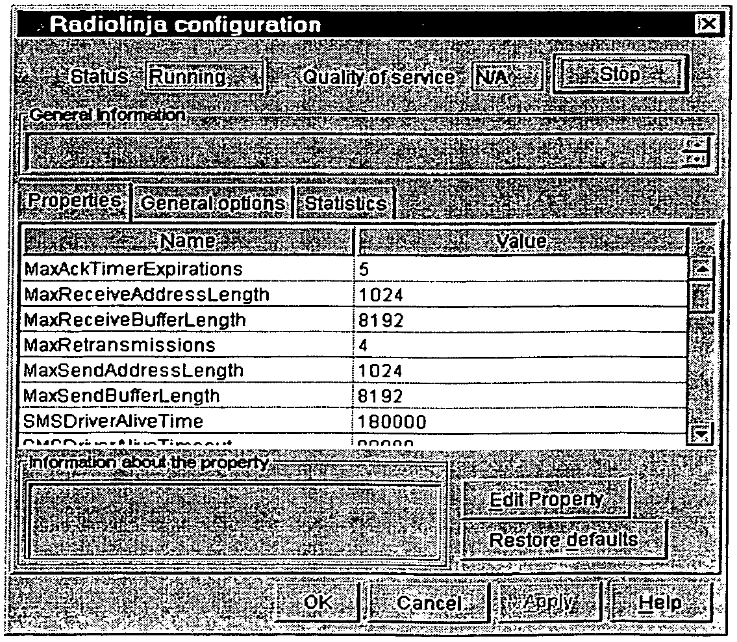
61
MgmtCntx
62
SendCntx
63
RecvCntx1
64
RecvCntx2
BA instance 1
51
BA instance 2
51

Fig. 5



Name	Type	Status	Quality of service
Radiolinja	Nokia_CIMD_10	stopped	N/A?

Fig. 6a



Status: Running

Quality of service: N/A

Name	Value
MaxAckTimerExpirations	5
MaxReceiveAddressLength	1024
MaxReceiveBufferLength	8192
MaxRetransmissions	4
MaxSendAddressLength	1024
MaxSendBufferLength	8192
SMSDriverAliveTime	180000

Fig. 6b

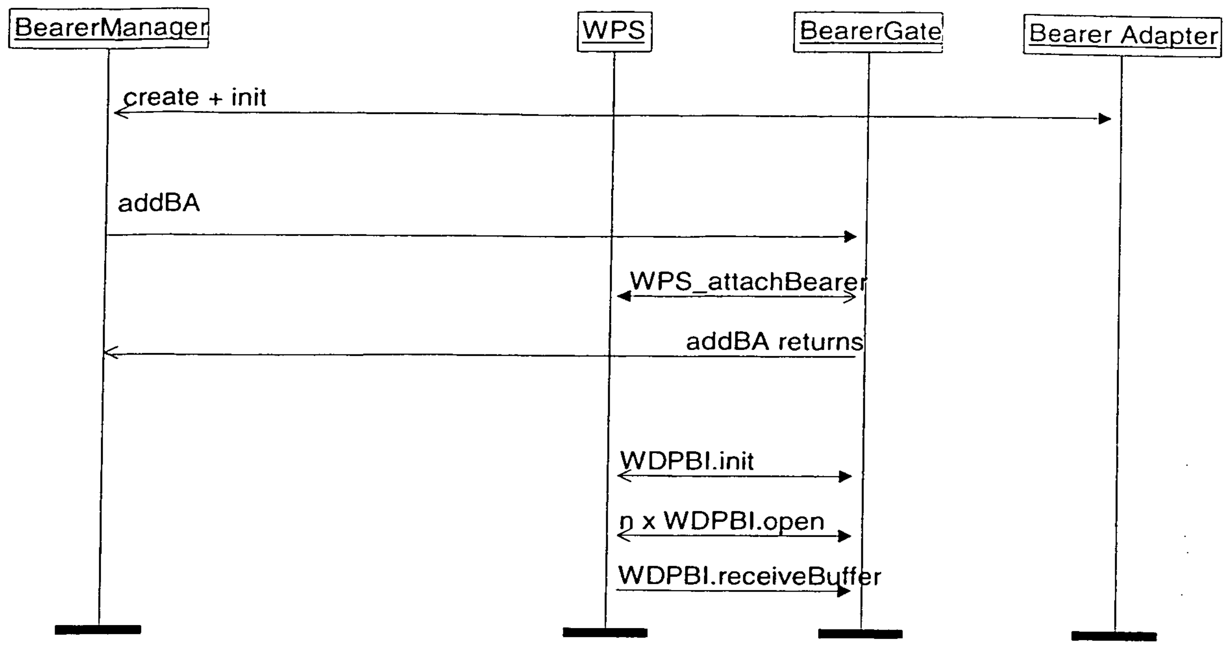

BearerManager
WPS
BearerGate
Bearer Adapter
create + init
addBA
WPS_attachBearer
addBA returns
WDPBI.init
n x WDPBI.open
WDPBI.receiveBuffer

Fig. 7a

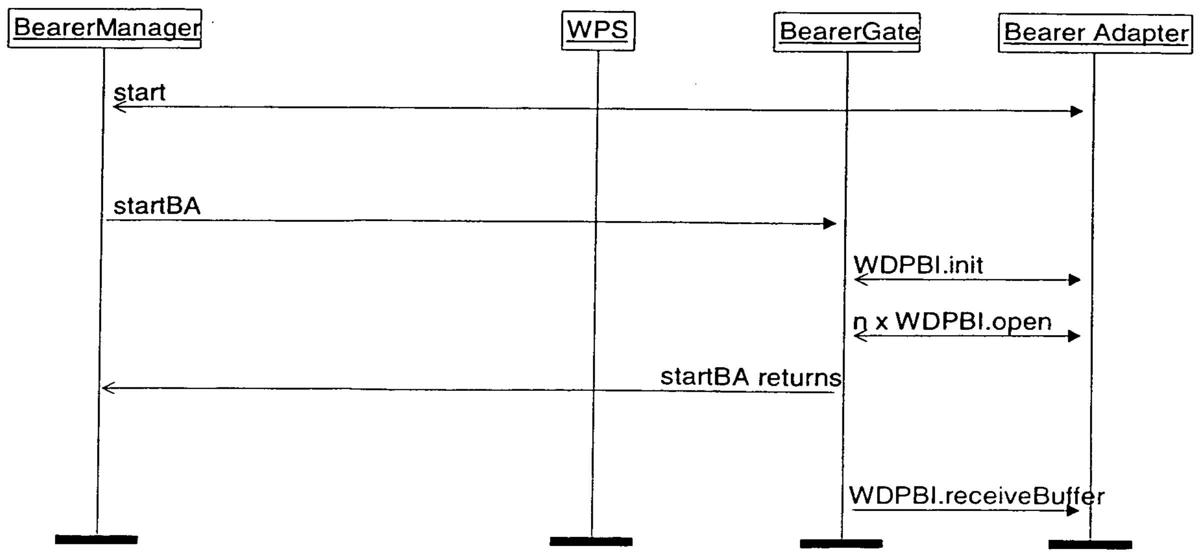

BearerManager
WPS
BearerGate
Bearer Adapter
start
startBA
WDPBI.init
n x WDPBI.open
startBA returns
WDPBI.receiveBuffer

Fig. 7b

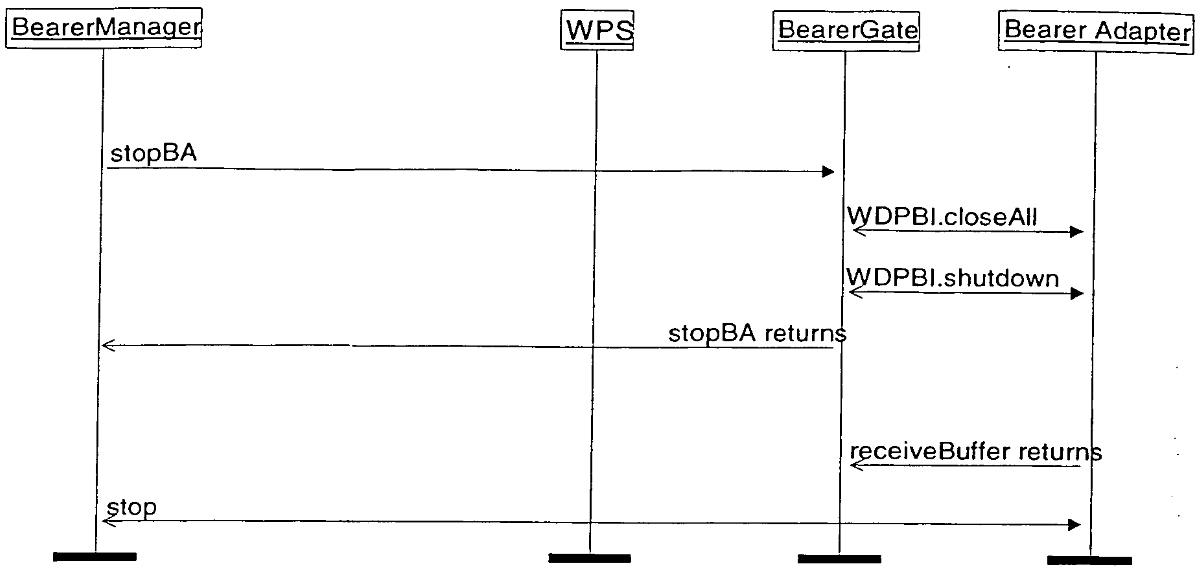

BearerManager
WPS
BearerGate
Bearer Adapter
stopBA
WDPBI.closeAll
WDPBI.shutdown
stopBA returns
receiveBuffer returns
stop

Fig. 7c

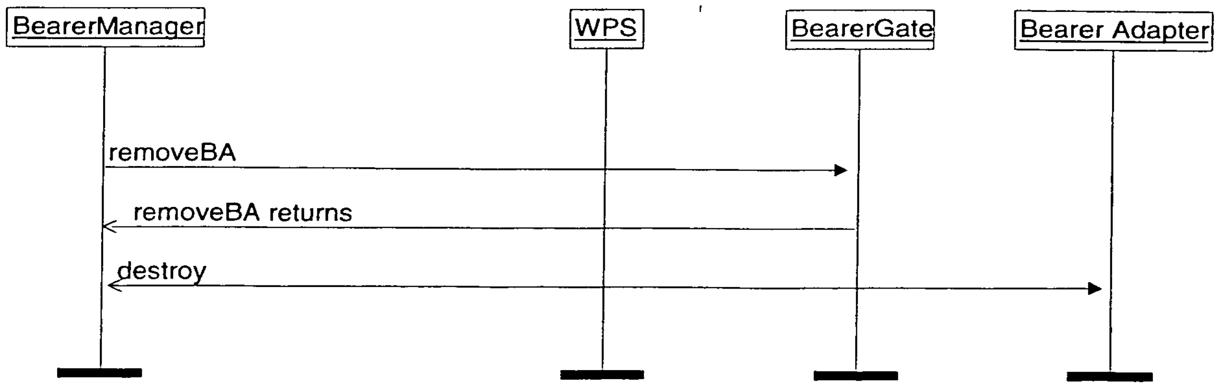

BearerManager
WPS
BearerGate
Bearer Adapter
removeBA
removeBA returns
destroy

Fig. 7d